Importing the data using Cypher:

Below script is used to create the Nodes:

**//Create customers**

LOAD CSV WITH HEADERS FROM' file:///customers.csv' AS row

Create (:customers{customerName:row.customerName,customerNumber:row.customerNumber,country:row.country,creditLimit:row.creditLimit})

**//Create payments**

LOAD CSV WITH HEADERS FROM' file:///payments.csv' AS row

Create (:payments{checkNumber:row.checkNumber,amount:row.amount,paymentDate:row.paymentDate })

**//Create employees**

LOAD CSV WITH HEADERS FROM' file:///employees.csv' AS row

Create (:employees{employeeNumber:row.employeeNumber,jobTitle:row.jobTitle })

**//Create offices**

LOAD CSV WITH HEADERS FROM' file:///offices.csv' AS row

Create (:offices{officeCode:row.officeCode,country:row.country})

**//Create orders**

LOAD CSV WITH HEADERS FROM' file:///orders.csv' AS row

Create (:orders{orderNumber:row.orderNumber,orderDate:row.orderDate,requiredDate:row.requiredDate,shippedDate:row.shippedDate,status\_name:row.status\_name})

**//Create orderdetails**

LOAD CSV WITH HEADERS FROM' file:///orderdetails.csv' AS row

Create (:orderdetails{order\_id:row.order\_id,quantityOrdered:row.quantityOrdered,priceEach:row.priceEach})

**//Create products**

LOAD CSV WITH HEADERS FROM' file:///products.csv' AS row

Create(:products{productCode:row.productCode,productName:row.productName,productLine:row.productLine,quantityInStock:row.quantityInstock,buyPrice:row.buyPrice,MSRP:row.MSRP})

**//Create productLines**

LOAD CSV WITH HEADERS FROM' file:///productlines.csv' AS row

Create (:productLines{p\_id:row.p\_id,productLine:row.productLine})

Below script is used to produce indexes on the nodes created to confirm their quick lookup when producing relationship.

CREATE INDEX ON :product(productCode);

CREATE INDEX ON :product(productName);

CREATE INDEX ON :productLines(p\_id);

CREATE INDEX ON :productLines(productLine);

CREATE INDEX ON :employees(employeeNumber);

CREATE INDEX ON :customers(customerNumber);

CREATE INDEX ON :customers(customerName);

CREATE CONSTRAINT ON (o:order) ASSERT o.orderNumber IS UNIQUE;

Below script is for creating relationship of orderdetails to products

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///ordersdetails.csv" AS row

MATCH (orderdetails:orderdetails {order\_id: row.order\_id})

MATCH (products:products {productCode: row.productCode})

MERGE (orderdetails)-[pu:PRODUCT]->(products)

ON CREATE SET pu.MSRP = toFloat(row.MSRP), pu.quantityOrdered = toFloat(row.quantityOrdered);

Below script is for creating relationship of orders to customers

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///orders.csv" AS row

MATCH (orders:orders {orderNumber: row.orderNumber})

MATCH (customers:customers {customerName: row.customerName})

MERGE (customer)-[:PLACE]->(orders);

Below script is for creating relationship of products to productLines

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///products.csv" AS row

MATCH (products:products {productCode: row.productCode})

MATCH (productLines:productLines {p\_id: row.p\_id})

MERGE (productLines)-[:DESCRIBES]->(products);

Below script is for creating relationship of customers to employees:

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///employees.csv" AS row

MATCH (customers:customers {customerNumber: row.customerNumber})

MATCH (employees:employees {employeeNumber: row.reportsTo})

MERGE (customers)-[:REPORTS\_TO]->(employees);

LOAD CSV FROM 'file:///products.csv' AS row

WITH row[0] AS productCode ,row[1] AS productName, row[2] AS productLines

RETURN productCode, productName, productLines

LIMIT 8

LOAD CSV WITH HEADERS FROM 'file:///orderdetails.csv' AS row

WITH (row.productCode) AS productCode, toInteger(row.order\_id) AS order\_id, toInteger(row.quantityOrdered) AS quantityOrdered

RETURN productCode, order\_id, quantityOrdered

LIMIT 8

USING PERIODIC COMMIT 500

LOAD CSV WITH HEADERS FROM 'file:///orderdetails.csv' AS row

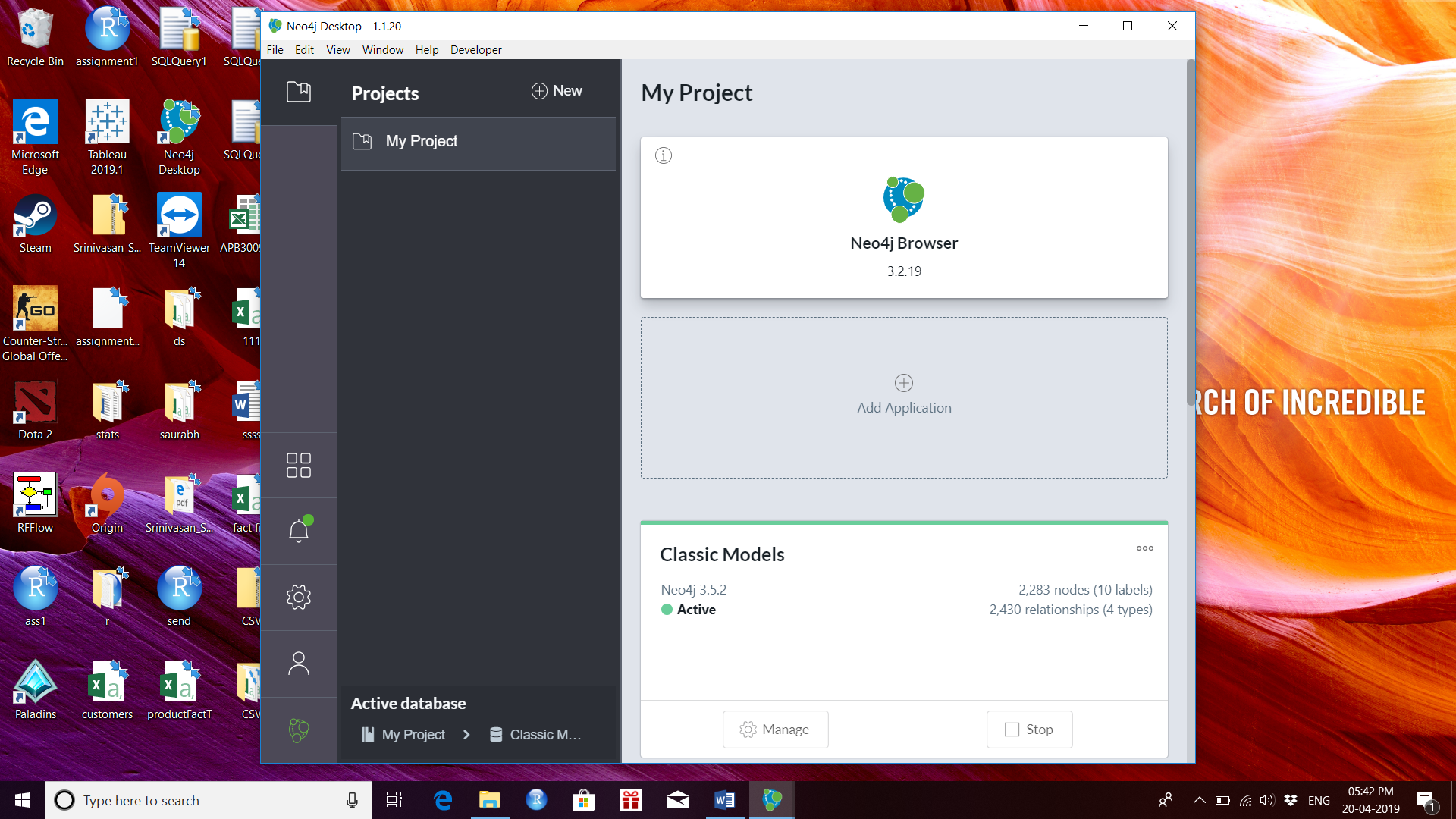
WITH (row.productCode) AS productCode, (row.order\_id) AS order\_id, (row.quantityOrdered) AS quantityOrdered

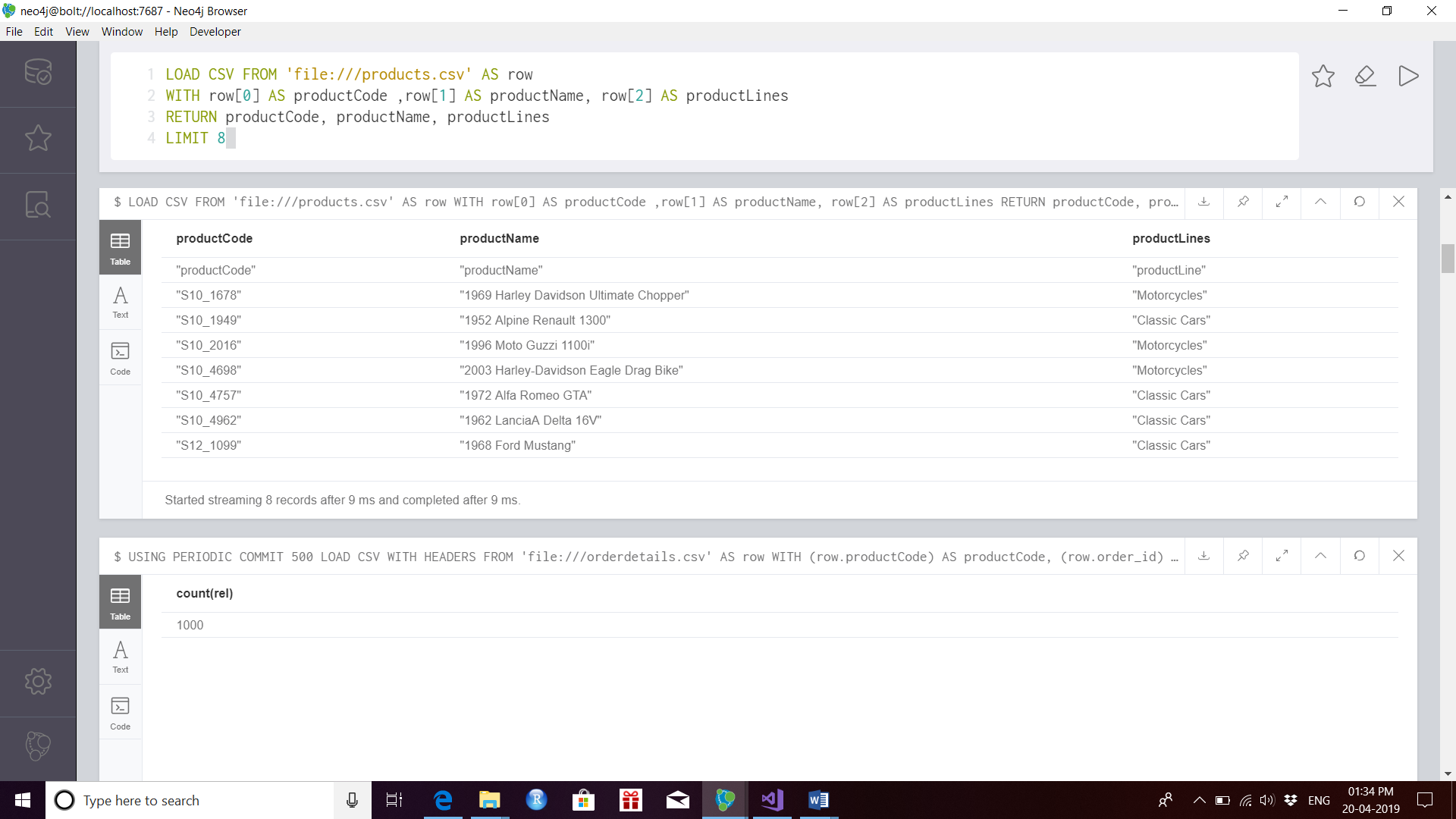
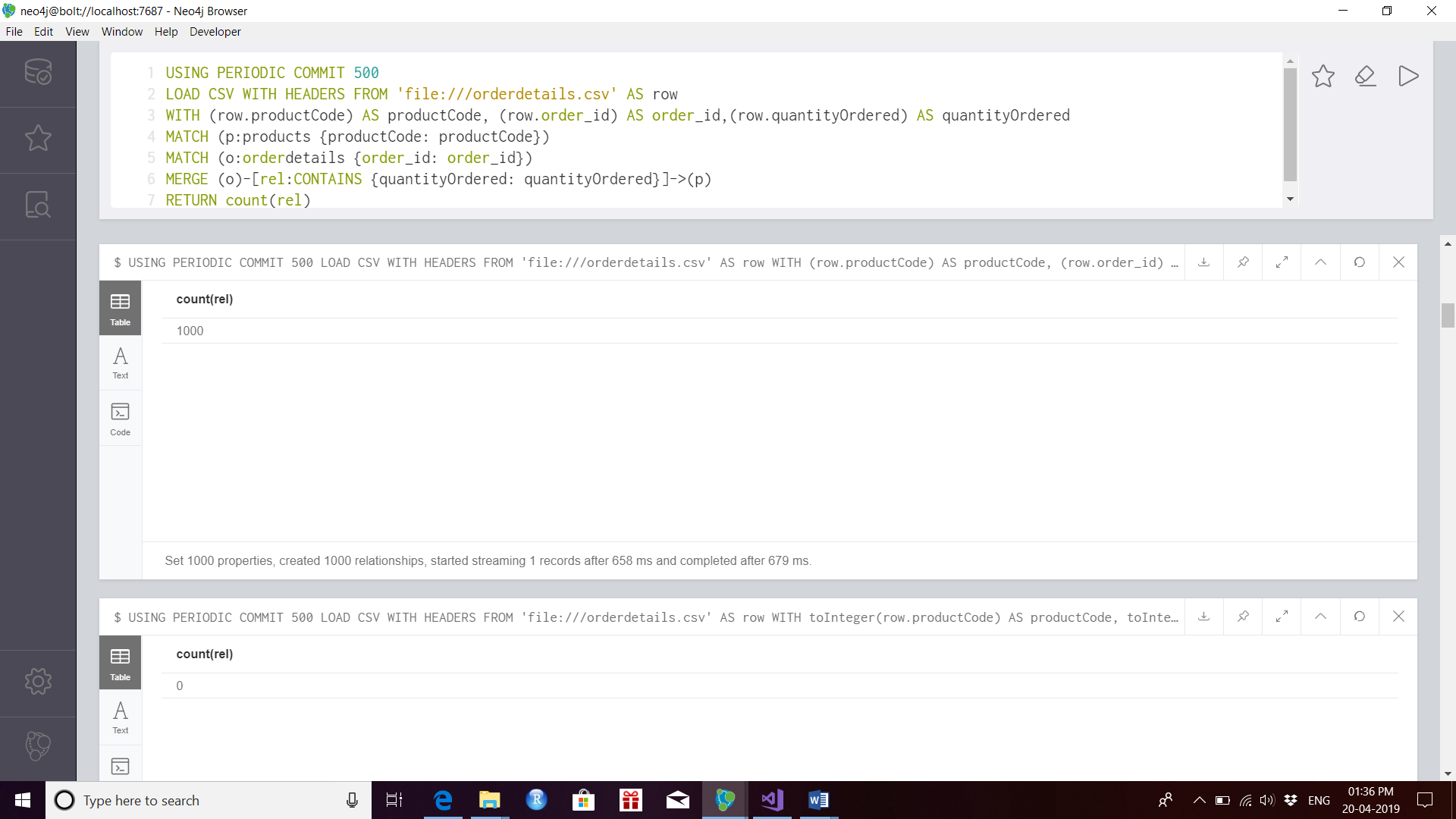
MATCH (p:products {productCode: productCode})

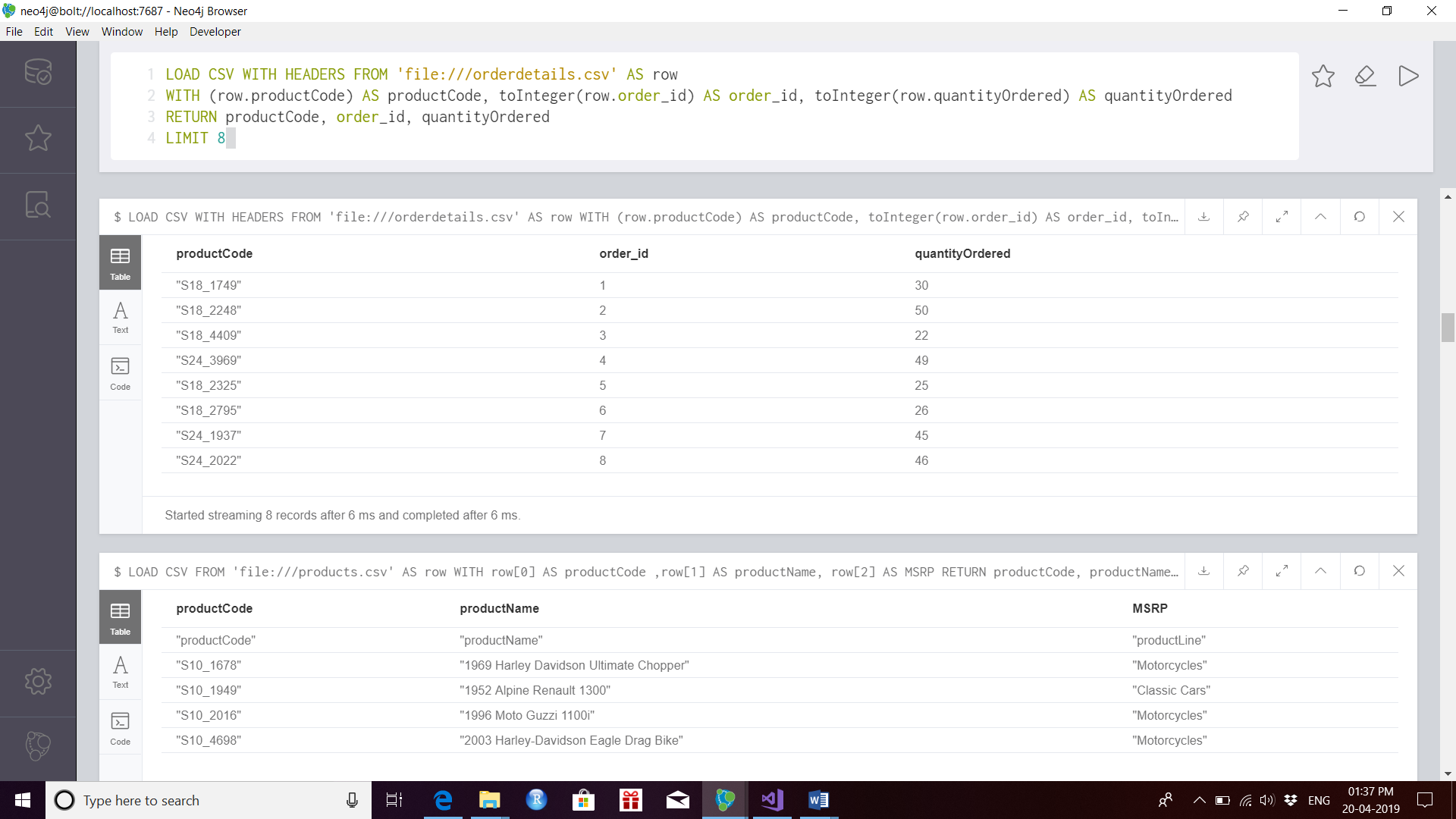
MATCH (o:orderdetails {order\_id: order\_id})

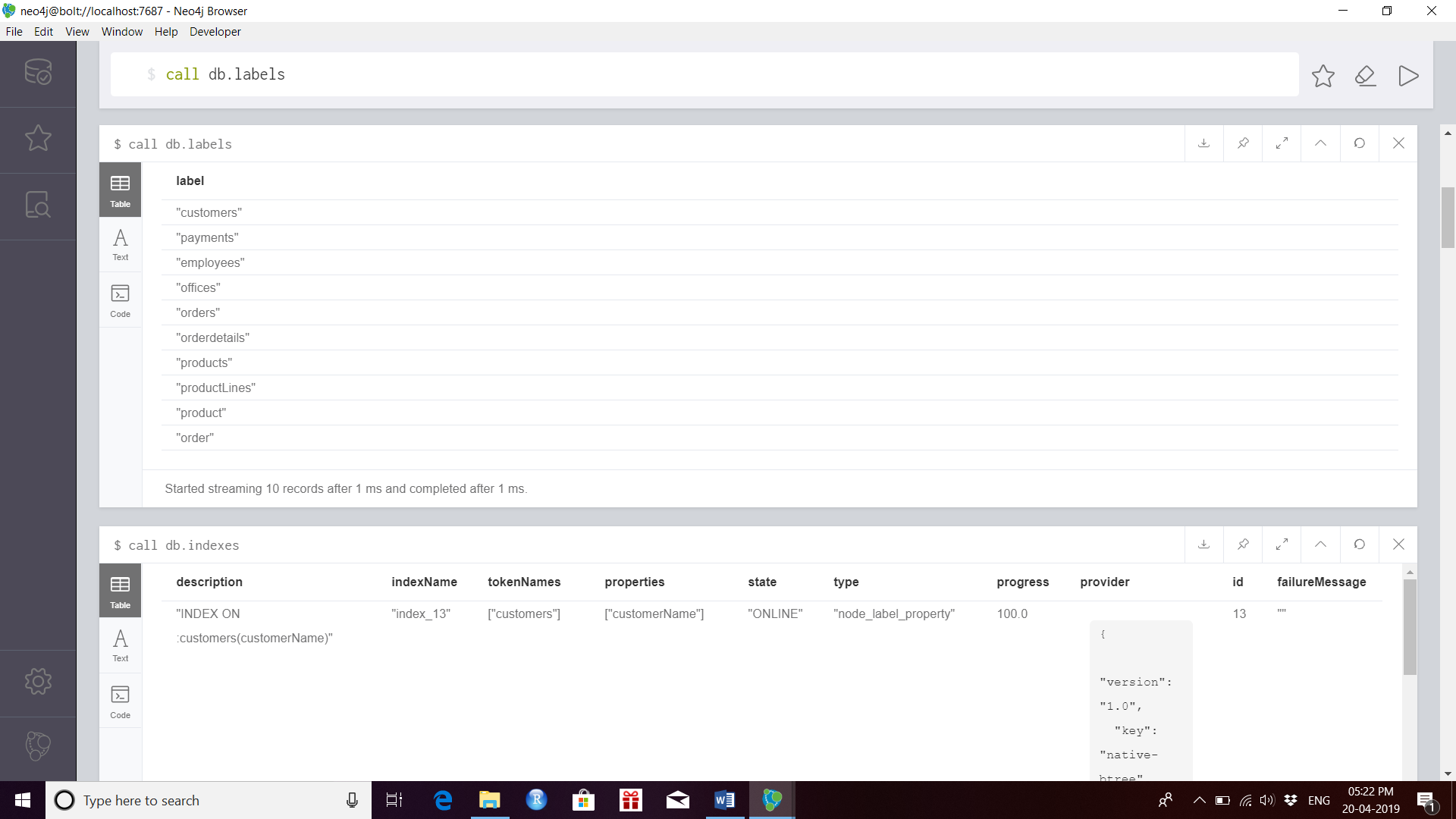
MERGE (o)-[rel:CONTAINS {quantityOrdered: quantityOrdered}]->(p)

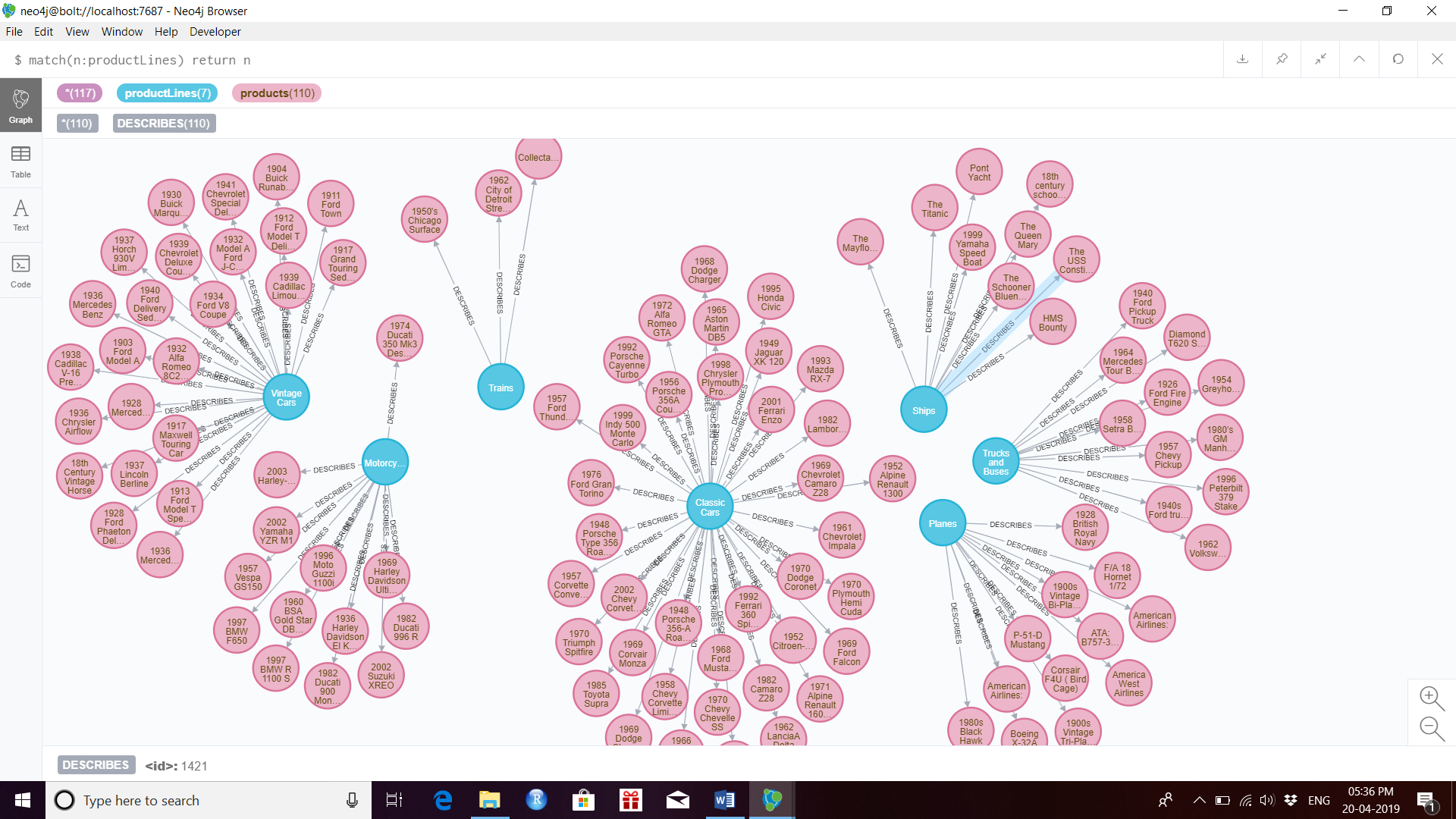
RETURN count(rel)

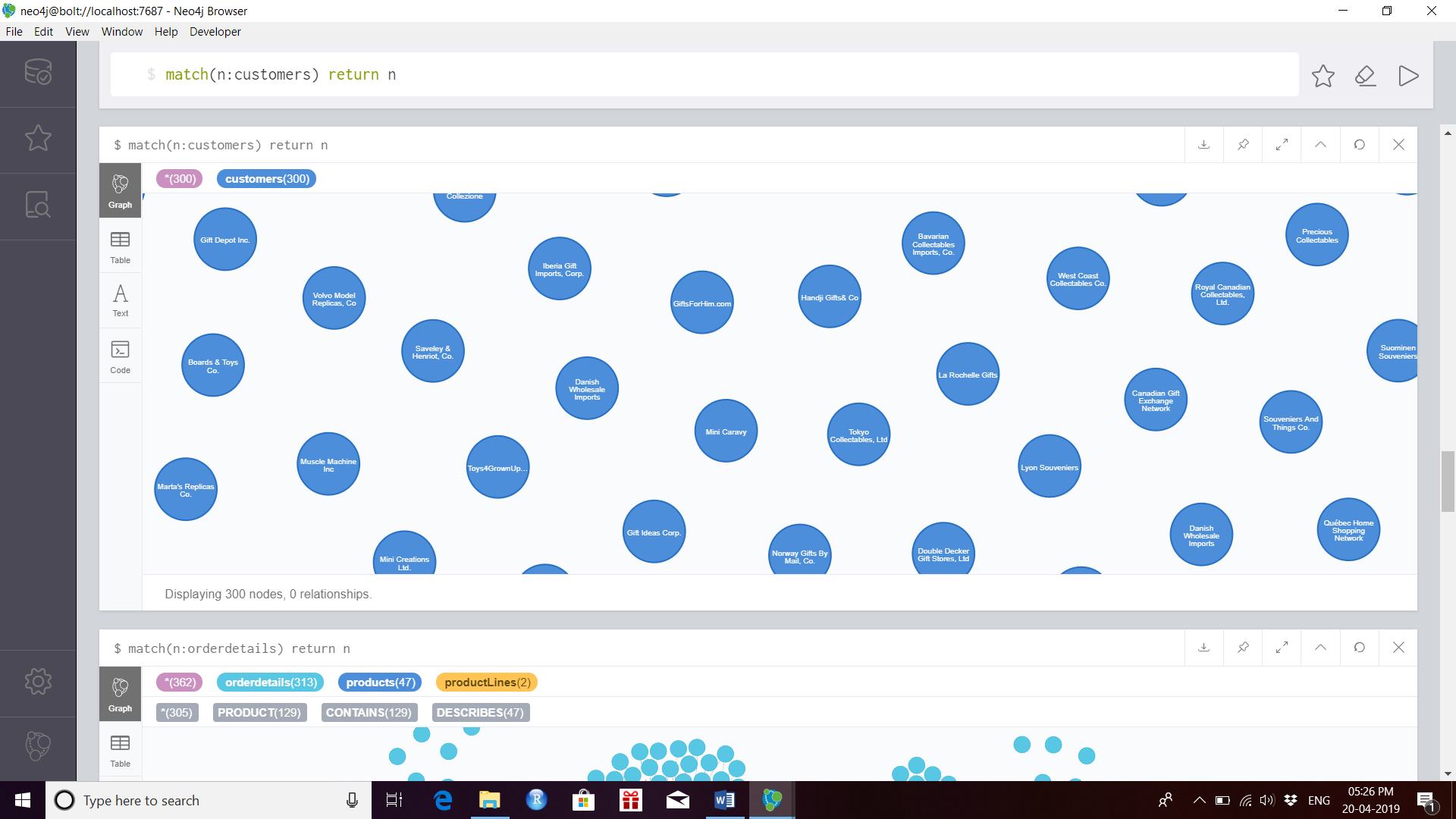


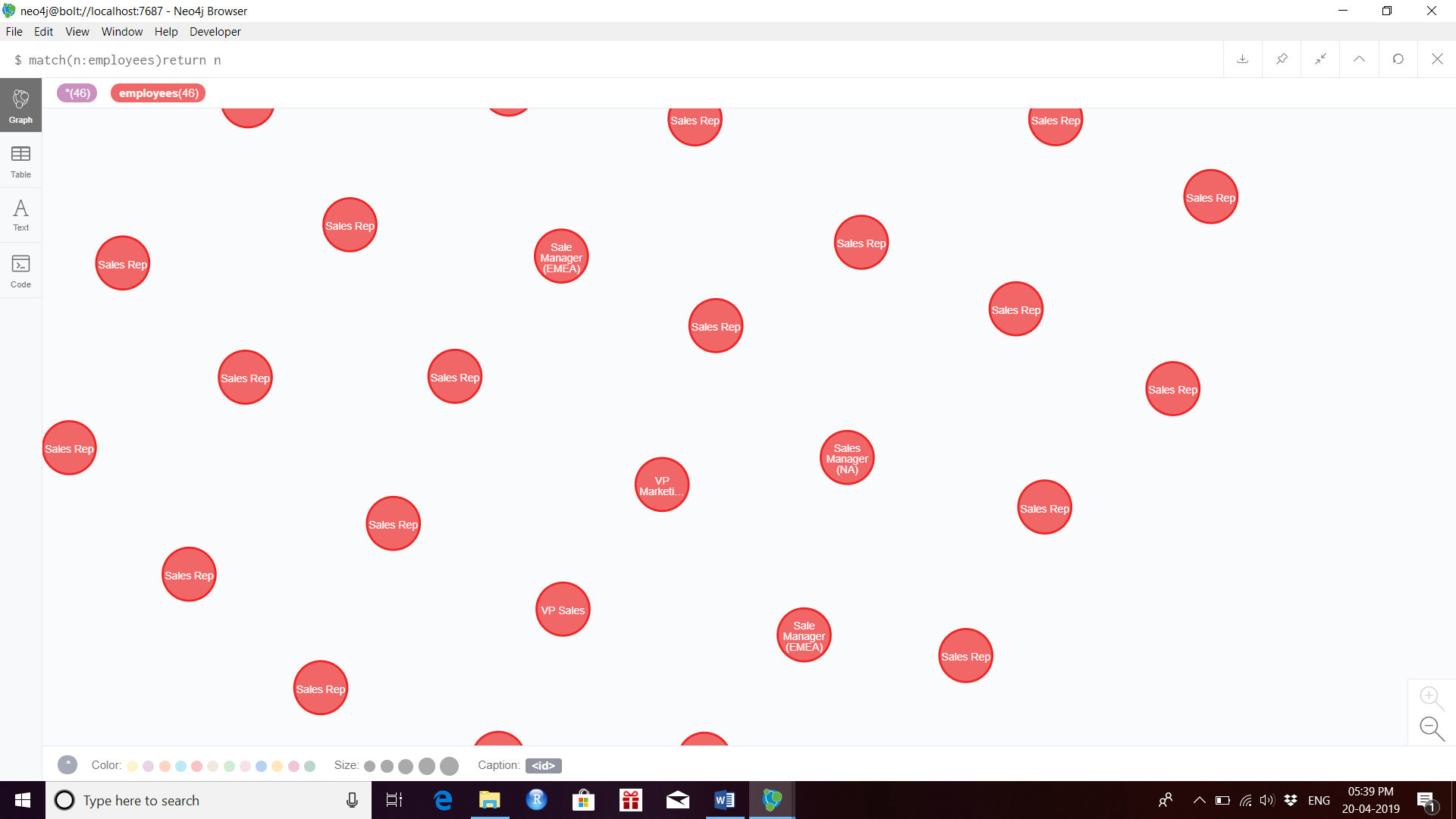


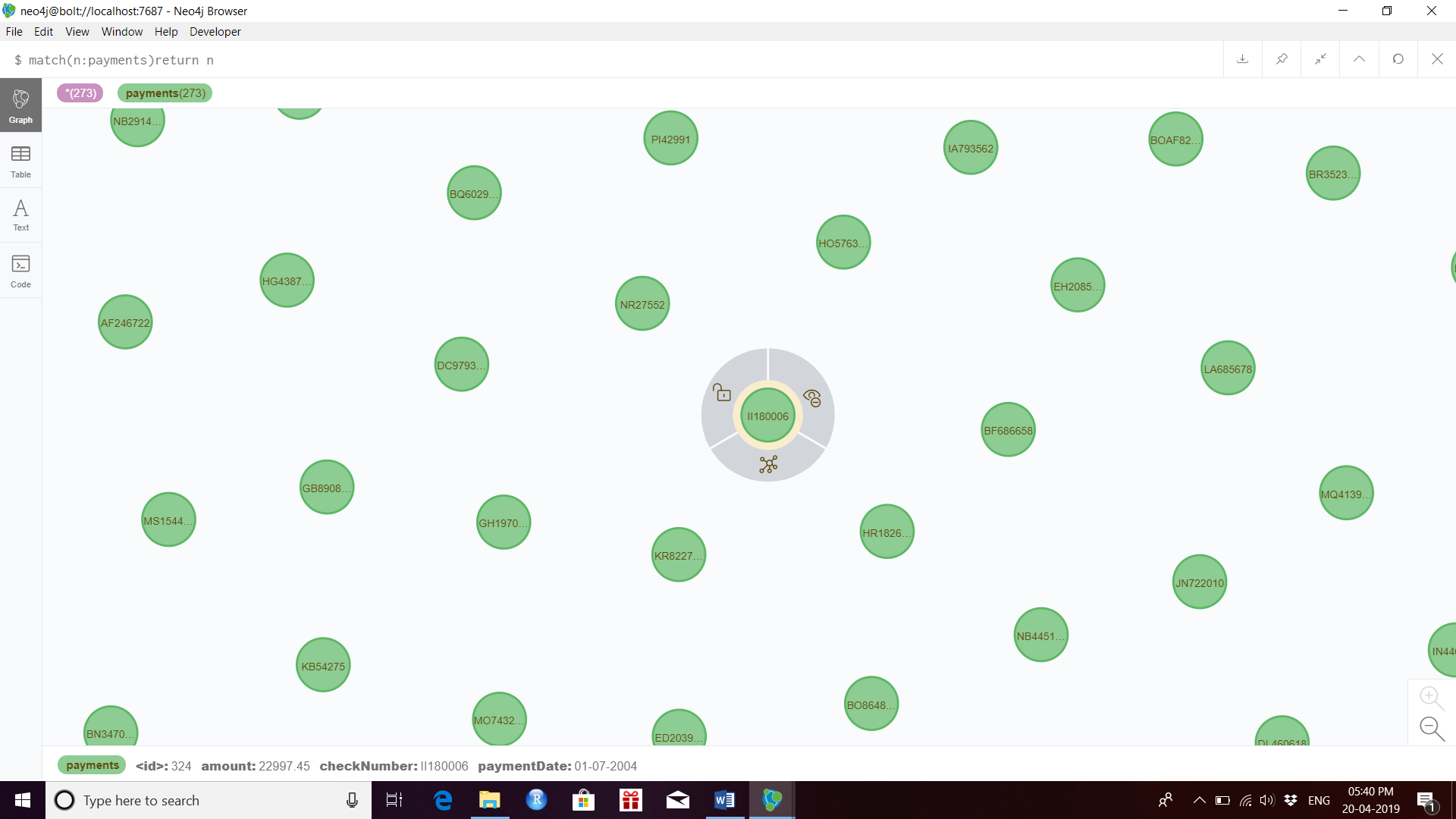


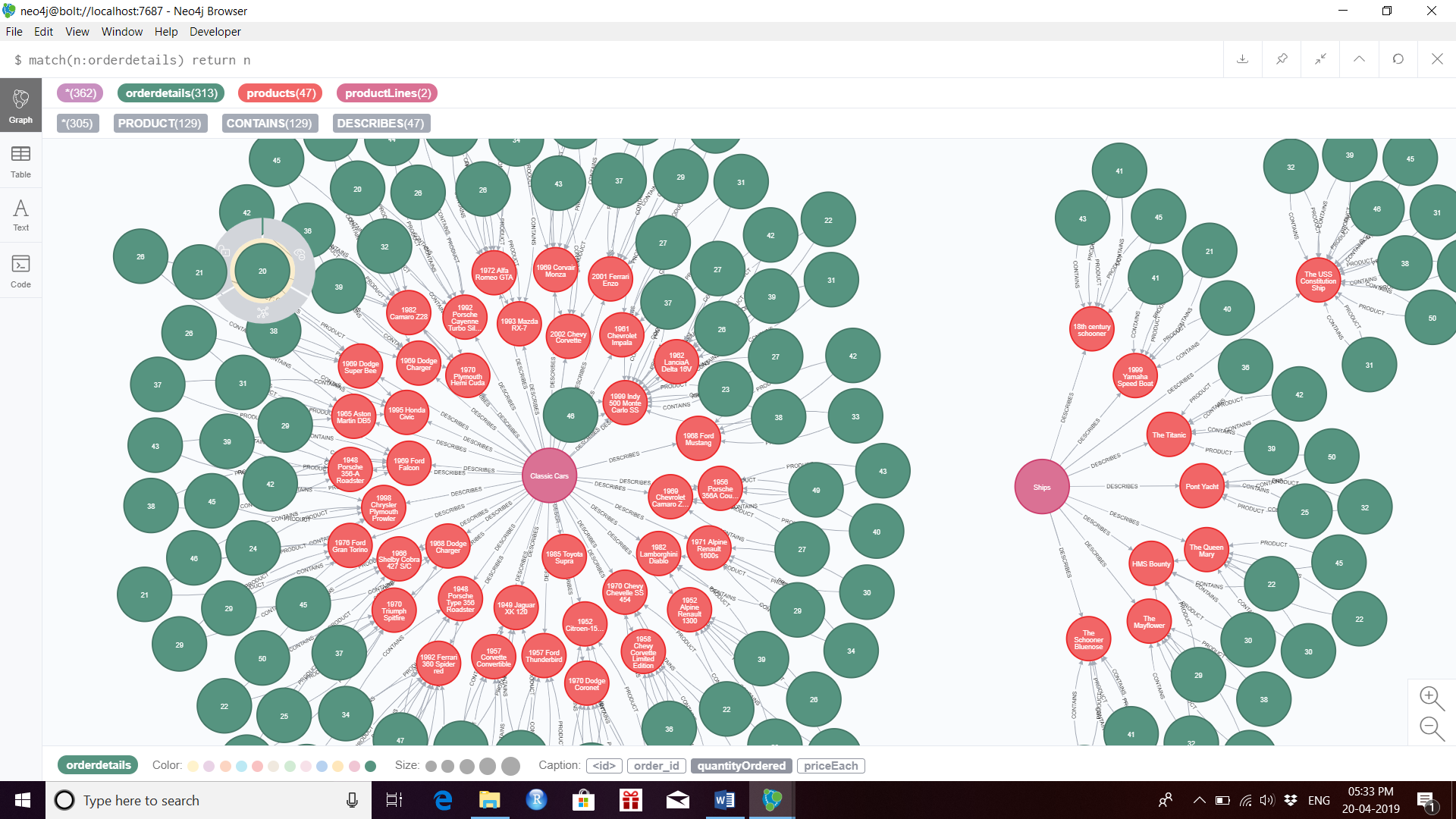


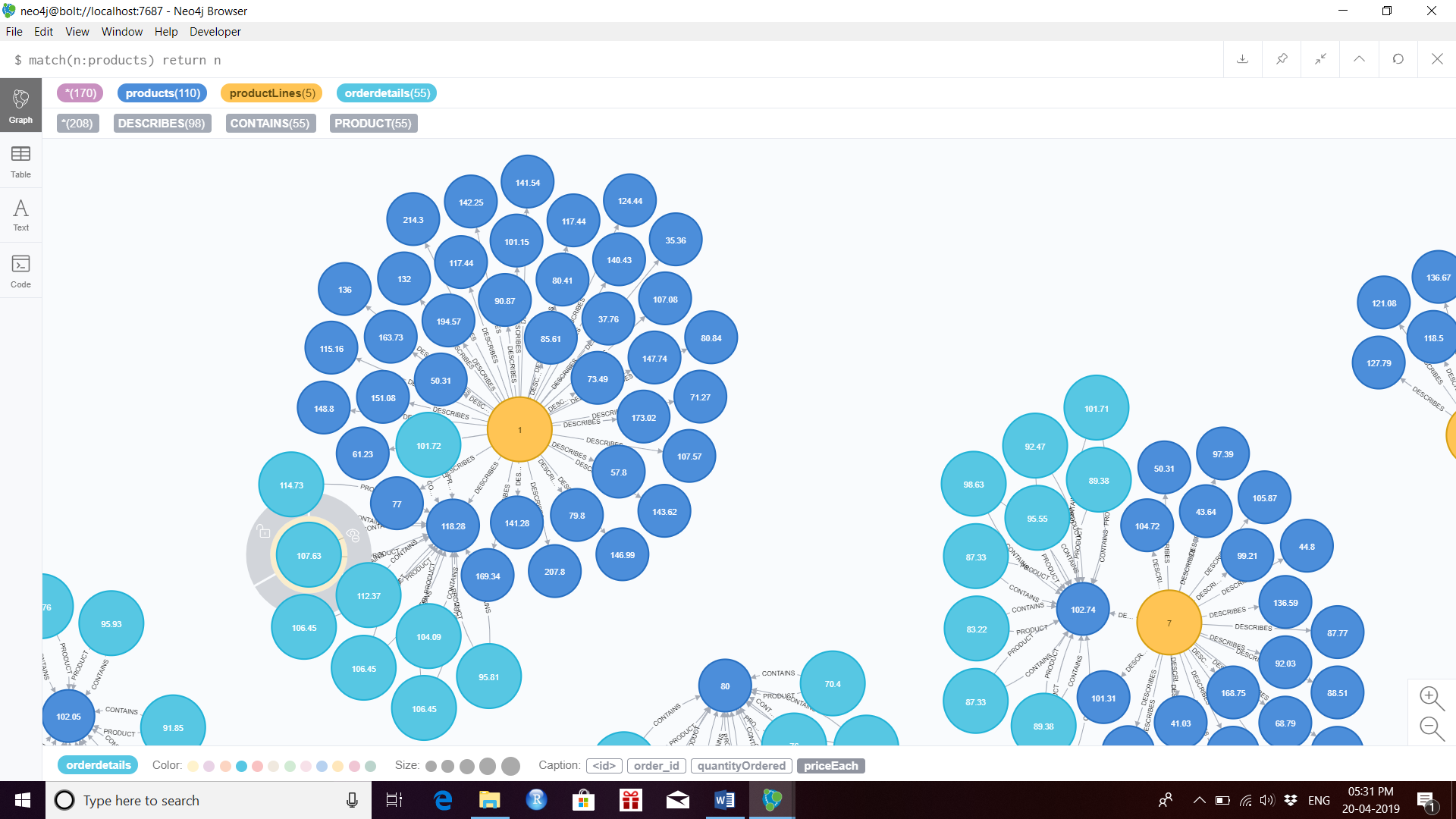












Creating relationship between employees to offices

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///employees.csv" AS row

MATCH (employees:employees {employeeNumber: row.employeeNumber})

MATCH (offices:offices {officeCode: row.officeCode})

MERGE (employees)-[:BELONGS\_TO]->(offices);

